图文 005、JVM的垃圾回收机制是用来干嘛的？为什么要垃圾回收？

5069 人次阅读 2019-07-05 07:00:00

详情 评论

**JVM的垃圾回收机制是用来干嘛的?**

**为什么要垃圾回收?**

---

**给大家推荐一套质量极高的Java面试训练营课程：**


**作者是中华石杉，石杉老哥是我之前所在团队的 Leader，骨灰级的技术神牛！**

大家可以点击下方链接，了解更多详情，并进行试听：

21天互联网Java进阶面试训练营（分布式篇）

---

推荐结束，正文开始

**目录：**



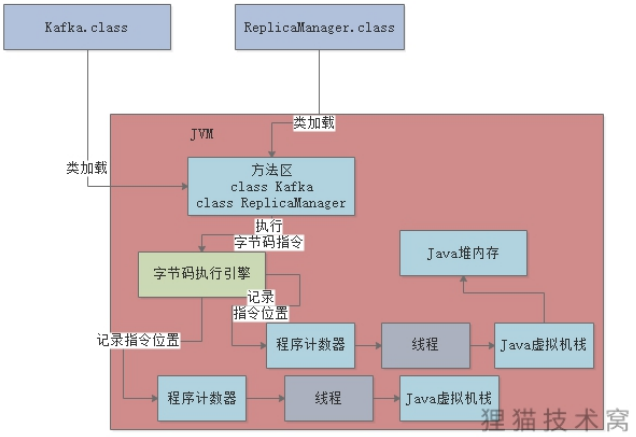
**1、前文回顾**

上一篇文章给大家分析了JVM中的几块内存区域分别都是干什么的，今天的文章就给大家初步介绍一下垃圾回收的概念。

但是今天的文章对垃圾回收不会切入过深，因为很多学习专栏的朋友都是一些初学者。

因此，咱们还是那句话，尽量用最通俗的语言配合大量手绘图，让大家初步了解垃圾回收到底是什么。

先来看一下昨天的一张图，回顾一下JVM中几块内存区域的作用。




大家脑子里一定要有一个会动的图，你的代码在运行的时候，起码有一个main线程会去执行所有的代码，当然也可能是你启动的别的线程。

然后线程执行时必须通过自己的程序计数器来记录执行到哪一个代码指令了

另外线程在执行方法时，为每个方法都得创建一个栈帧放入自己的Java虚拟机栈里去，里面有方法的局部变量。

最后就是代码运行过程中创建的各种对象，都是放在Java堆内存里的。

结合上面的大图看一看，相信大家一定就明白是怎么回事了，大家对JVM的运行原理也应该都有了一个初步的理解和把握。

**2、对象的分配与引用**

现在我们假设有下面一段代码，大概意思你可以理解为通过“loadReplicasFromDisk”方法的执行，去磁盘上加载需要的副本数据

然后通过“ReplicaManager”对象实例完成了这个操作。

代码如下所示：

```
public class Kafka {

        public static void main(String[] args) {
             loadReplicasFromDisk();
        }

        private static void loadReplicasFromDisk() {
              ReplicaManager replicaManager = new ReplicaManager();
              replicaManager.load();
        }

}
```

结合我们之前理解过的JVM运行原理，一起通过动态的图来拆解一下上述代码的运行流程。

首先一个main线程肯定会来执行main()方法里的代码

main线程自己是有一个Java虚拟机栈的，他会把main()方法的栈帧压入Java虚拟机栈

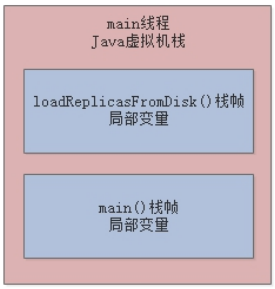
如下图所示：




接着main()方法里调用了loadReplicasFromDisk()方法

那么就会创建loadReplicasFromDisk()方法的栈帧，压入main线程的Java虚拟机栈里去

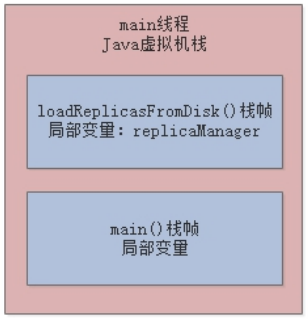
这个过程如下图：






此时发现在loadReplicasFromDisk()方法里，有一个“repliaManager”变量，那么就会在loadReplicasFromDisk()方法对应的栈帧里，放入一个“repliaManager”变量。

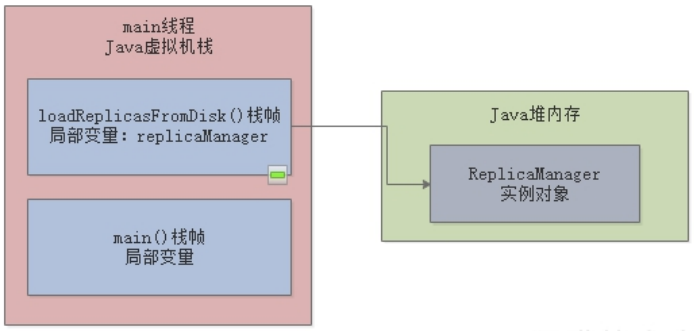
继续看下图：






接着发现在代码里创建了一个“ReplicaManager”类的实例对象，此时就会在Java堆内存中分配这个实例对象的内存空间。

同时，让loadReplicasFromDisk()方法的栈帧内的“replicaManager”局部变量去指向那个Java堆内存里的ReplicaManager实例对象，大家看下图：






接下来，就会执行通过“replicaManager”局部变量引用的“ReplicaManager”实例对象去执行他的load()方法，去完成我们实现的业务逻辑。

好，到这里为止，其实都是上篇文章讲解过的知识，我们就是重新串联了一遍，相信大家都很好理解。

**3、一个方法执行完毕之后会怎么样?**

接着大家来回顾一下上面的代码。

```
public class Kafka {

     public static void main(String[] args) {
          loadReplicasFromDisk();
     }

     private static void loadReplicasFromDisk() {
           ReplicaManager replicaManager = new ReplicaManager();
           replicaManager.load();
     }

}
```

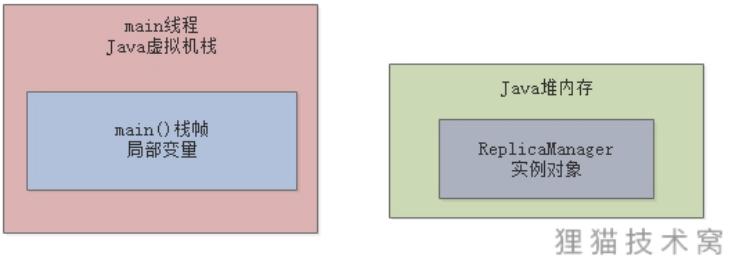
其实目前的图我们已经表述到了“replicaManager.load()”这行代码这里

那么现在有个问题，如果这行代码执行结束了，此时会怎么样?

大家还记得之前文章说过，一旦方法里的代码执行完毕，那么方法就执行完毕了，也就是说loadReplicasFromDisk()方法就执行完毕了。

一旦你的loadReplicasFromDisk()方法执行完毕，此时就会把loadReplicasFromDisk()方法对应的栈帧从main线程的Java虚拟机栈里出栈

如下图所示:






此时一旦loadReplicasFromDisk()方法的栈帧出栈，那么大家会发现那个栈帧里的局部变量，“replicaManager”，也就没有了。

也就是说，没有任何一个变量指向Java堆内存里的“ReplicaManager”实例对象了。

**4、我们创建的Java对象其实都是占用内存资源的**

核心点来了，此时大家发现了，Java堆内存里的那个“ReplicaManager”实例对象已经没有人引用他了

这个对象实际上已经没用了，该干的事儿都干完了，现在你还让他留在内存里干啥呢？

大家要知道，内存资源是有限的。

一般来说，我们会在一台机器上启动一个Java系统，机器的内存资源是有限的，比如就4个G的内存

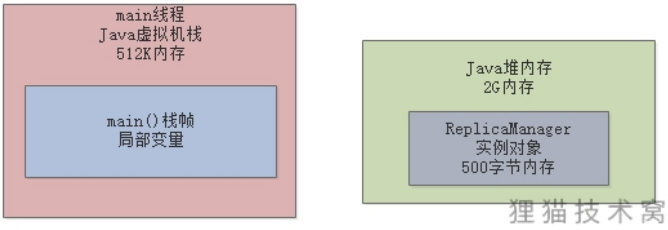
然后我们启动的Java系统本质就是一个JVM进程，他负责运行我们的系统的代码，这个之前都解释过了。

那么这个JVM进程本身也是会占用机器上的部分内存资源，比如占用2G的内存资源。

那么我们在JVM的Java堆内存中创建的对象，其实本质也是会占用JVM的内存资源的，比如“ReplicaManager”实例对象，会占用500字节的内存。

所以大家看到这里，心中应该无比明白的一个核心点：我们在Java堆内存里创建的对象，都是占用内存资源的，而且内存资源有限。

大家看下面的图，感受会深一点。




**5、不再需要的那些对象应该怎么处理？**

继续思考上面的图，既然“ReplicaManager”对象实例是不需要使用的，已经没有任何方法的局部变量在引用这个实例对象了，而且他还空占着内存资源，那么我们应该怎么处理呢？

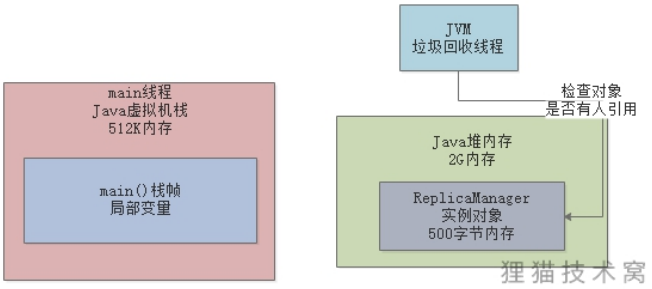
答案呼之欲出：**JVM的垃圾回收机制**

JVM本身是有垃圾回收机制的，他是一个后台自动运行的线程

你只要启动一个JVM进程，他就会自带这么一个垃圾回收的后台线程。

这个线程会在后台不断检查JVM堆内存中的各个实例对象

还是给大家画一张图，来看看这个过程：

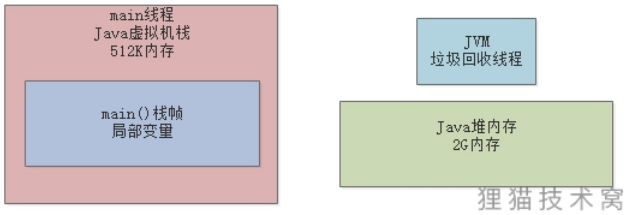



如果某个实例对象没有任何一个方法的局部变量指向他，也没有任何一个类的静态变量，包括常量等地方在指向他。

那么这个垃圾回收线程，就会把这个没人指向的“ReplicaManager”实例对象给回收掉，从内存里清除掉，让他不再占用任何内存资源。

这样的话，这些不再被人指向的对象实例，即JVM中的“**垃圾**”，就会定期的被后台垃圾回收线程清理掉，不断释放内存资源

大家看下图：




到此为止，相信大家跟上文章思路一路看下来，就很清晰明了。到底什么是JVM中的“垃圾”？什么又是JVM的“垃圾回收”！

**6、本文小结**

不知不觉，第一周的文章都更新完毕了，希望大家温故而知新。

既然是付费来学习知识的，一定要对自己负责，坚持把每篇文章多看几遍，把知识吃透。

这周，我们为了照顾很多JVM的小白同学，从0起步，用最通俗易懂的语言和一步一图的方式，把你写好的Java代码如何通过JVM运行起来的核心原理都讲清楚了。

目前为止，大家应该对JVM的核心运行流程、JVM的类加载机制、JVM的内存区域以及垃圾回收机制都有一个初步的了解。

很多对JVM有一定了解的朋友一定会说：这些内容都很简单，有没有深入点的干货?

**有！**但是请不要着急，我们的专栏是兼顾各种基础的朋友，所以需要循序渐进，从浅入深。

比如JVM垃圾回收机制，就会在第三周详细讲解JVM的各种垃圾回收的细节

所以请大家稍安勿躁，一步一步来，如果有一定基础的同学，就当做复习一遍。

马上周末就是第一周的作业和答疑集锦了，大家周末记得完成作业，同时看一下每周最新的精华答疑集锦，从答疑中也能学习到一些知识。

**7、希望大家多帮忙宣传和推广**

专栏刚上线的时候，是拜托我的好朋友在公众号里宣传的，现在他还在一直坚持帮我们宣传，非常的感谢他。

不过当时刚上线，我们一篇文章都没更新，很多朋友对我们的文章质量是没有了解的，所以可能会犹豫不决要不要购买

不过经过一周文章下来，相信很多一路跟下来的朋友，都对这个专栏质量有了自己的了解和认可。

我和即将发布专栏的几位朋友都长年工作于一线大厂，比如阿里、百度、美团，等等，负责过多个大型系统的架构设计。

之所以聚在狸猫技术窝这个平台开设专栏，初衷就是为了帮助国内广大的java工程师，解决大家在面试中、工作中的一些痛点问题。

现在网上很多技术资料，但是鱼龙混杂，底子薄弱的朋友很难辨别其内容优劣，很容易被带跑偏。

所以我们几个朋友才希望通过自己的思路，来做一些高质量的专栏，将我们多年一线工作经验浓缩精华，传授给大家，也是为国内IT界尽一点绵薄之力。

但是由于刚开始做，很多人不了解我们，所以还是希望购买我们专栏的朋友，如果觉得我们的内容不错，帮忙多宣传一下。

大家可以根据下面步骤生成自己的海报，推荐给朋友，分享技术的同时还能获得一点收益。

最后，感谢大家为我们坚持长期推出更多好的专栏做出的支持！

**8、昨日思考题解答**

我们回到文章中，昨天给了一个思考题：我们创建的那些对象，到底在Java堆内存里会占用多少内存空间呢？

这个其实很简单，一个对象对内存空间的占用，大致分为两块：

一个是对象自己本身的一些信息

一个是对象的实例变量作为数据占用的空间

比如对象头，如果在64位的linux操作系统上，会占用16字节，然后如果你的实例对象内部有个int类型的实例变量，他会占用4个字节，如果是long类型的实例变量，会占用8个字节。如果是数组、Map之类的，那么就会占用更多的内存了。

另外JVM对这块有很多优化的地方，比如补齐机制、指针压缩机制，这块东西我们不会单独拿出来讲，因为比较复杂，而且暂时对大家还不是很需要。

其实我信奉的一个道理，就是用案例实战来说话，引出很多技术和知识点的讲解，所以很多类似的知识，我会在后期大量的案例中去分析，需要的时候再学，效果最好。

这里就是先科普一下，让大家有一个基本的概念，如果有兴趣可以百度资料自己查阅。

**8、今日思考题**

既然今天提到了Java堆内存里的对象会被回收掉，那么加载到方法区的类会被垃圾回收吗？什么时候被回收？为什么呢？

大家可以思考一下这个问题，下周一的文章里会给出解答。

**End**



## 常见问题解答：

**一、 如何生成自己的分享海报并获取返现？**

**方式1**：

点击文章右上角**邀请好友**（如下图），生成自己的专属海报。

将海报发送给好友或分享朋友圈，朋友通过扫描你分享的海报购买课程，你将**获取返现24元**，可在个人中心中提现：

**累计邀请30人**，你将升级为高级推广员，此后每成功邀请一位朋友，返现翻倍。换句话说，从第31人开始，每成功邀请一位朋友，你将**获取返现48元**